Christopher Morgan

CS478 : Brother Christophe

Sep 20, 2013

A. k-Nearest Neighbors.

**1. Design a simple distance metric for this space. Briefly justify your choice.**

D =

Such that:

|  |  |  |
| --- | --- | --- |
| Abbreviation | Attribute Name | Attribute Representation |
| 0 | Outlook | 0 = Rainy  1 = Overcast  2 = Sunny |
| T | Temperature | 0 = Cool  1 = Mild  2 = Hot |
| H | Humidity | 0 = Normal  1 = Hot |
| W | Wind | 0 = Weak  1 = Strong |

Justification:

The thought here is that even though the data is nominal it has a very inherent ordinal, if not continuous feel. Outlook has an arguable scale to it. Conditions are improving as one walks from rainy to overcast to sunny. Overcast is less rainy and more sunny, where sunny is farther away from rainy than overcast, and sunny is less overcast. Therefore, the distance between these items holds, as well as their ordering. It is also reasonable that the ratios between these elements hold. With humidity and wind the argument is simply the existence of one or the other, either one or not. This means it is plausible that a numeric representation can work here for humidity and wind as well.

With this reasonable assumption that each attribute can be represented numerically, a Euclidian distance function should give a distance measure between each instance.

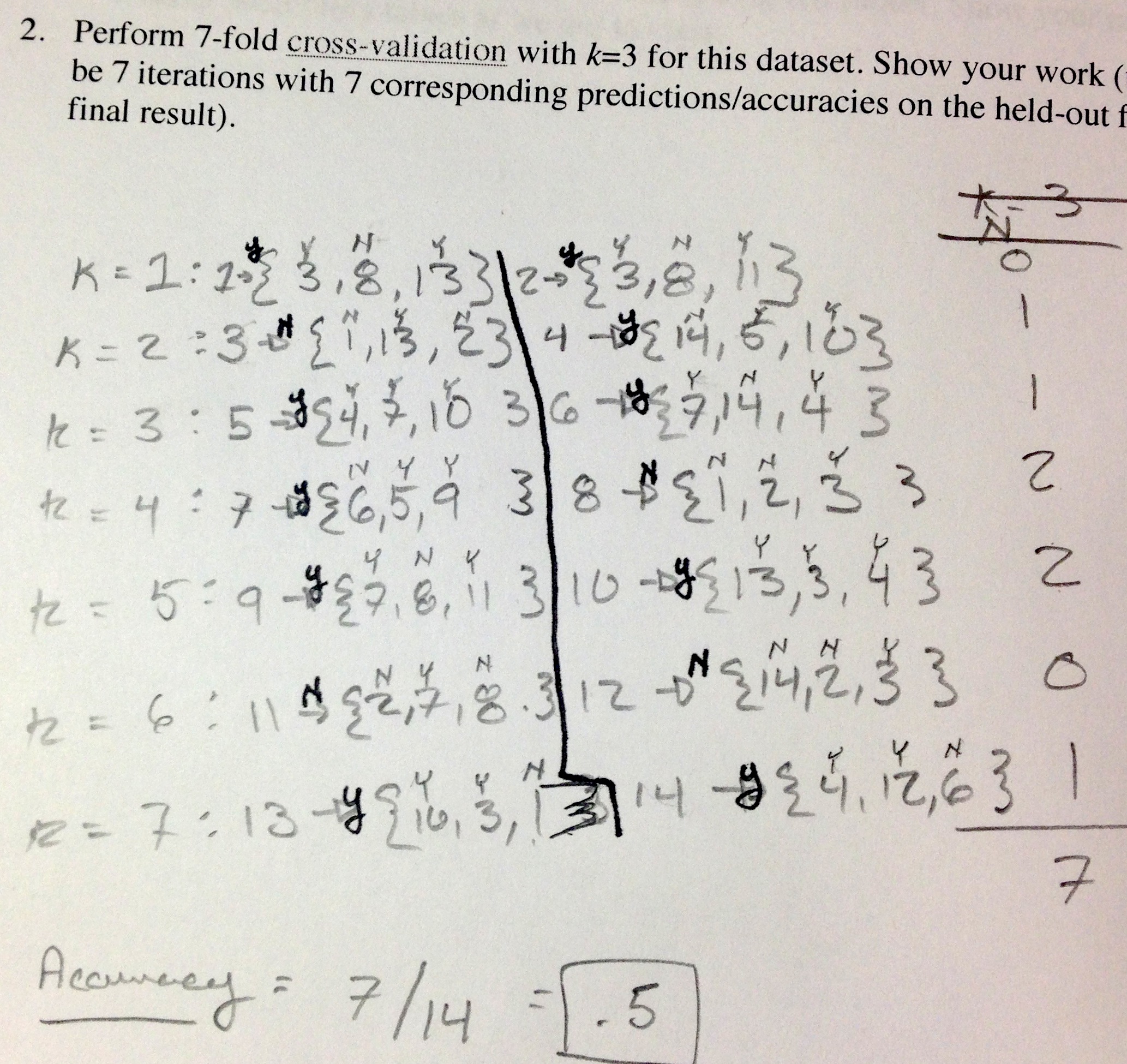
**2. Perform 7-fold cross-validation with k=3 for this dataset. Show your work (there should be 7 iterations with 7 corresponding predictions/accuracies on the held-out folds, and a final result).**

|  |  |  |
| --- | --- | --- |
| Run (Instances Used) | Number Correct (of 2) | Accuracy of Run |
| S1 (1,2) | 0 | 0 |
| S2 (3,4) | 1 | .5 |
| S3 (5,6) | 1 | .5 |
| S4 (7,8) | 2 | 1 |
| S5 (9,10) | 2 | 1 |
| S6 (11,12) | 0 | 0 |
| S7 (13,14) | 1 | .5 |
| Total | **7/14** | **.5** |

I used the following table, which was generated by the code found in the appendix. The table is a distance table, representing the distance from any instance to any other instance.

|  |
| --- |
| 0.00 | 1.00 | 1.00 | 2.24 | 3.00 | 3.16 | 2.65 | 1.00 | 2.24 | 1.73 | 1.73 | 1.73 | 1.41 | 2.45 | |
| 1.00 | 0.00 | 1.41 | 2.45 | 3.16 | 3.00 | 2.45 | 1.41 | 2.45 | 2.00 | 1.41 | 1.41 | 1.73 | 2.24 | |
| 1.00 | 1.41 | 0.00 | 1.41 | 2.45 | 2.65 | 2.45 | 1.41 | 2.45 | 1.41 | 2.00 | 1.41 | 1.00 | 1.73 | |
| 2.24 | 2.45 | 1.41 | 0.00 | 1.41 | 1.73 | 2.00 | 2.00 | 2.45 | 1.41 | 2.45 | 1.41 | 1.73 | 1.00 | |
| 3.00 | 3.16 | 2.45 | 1.41 | 0.00 | 1.00 | 1.41 | 2.45 | 2.00 | 1.41 | 2.45 | 2.00 | 2.24 | 1.73 | |
| 3.16 | 3.00 | 2.65 | 1.73 | 1.00 | 0.00 | 1.00 | 2.65 | 2.24 | 1.73 | 2.24 | 1.73 | 2.45 | 1.41 | |
| 2.65 | 2.45 | 2.45 | 2.00 | 1.41 | 1.00 | 0.00 | 2.00 | 1.41 | 1.41 | 1.41 | 1.41 | 2.24 | 1.73 | |
| 1.00 | 1.41 | 1.41 | 2.00 | 2.45 | 2.65 | 2.00 | 0.00 | 1.41 | 1.41 | 1.41 | 1.41 | 1.73 | 2.24 | |
| 2.24 | 2.45 | 2.45 | 2.45 | 2.00 | 2.24 | 1.41 | 1.41 | 0.00 | 1.41 | 1.41 | 2.00 | 2.24 | 2.65 | |
| 1.73 | 2.00 | 1.41 | 1.41 | 1.41 | 1.73 | 1.41 | 1.41 | 1.41 | 0.00 | 1.41 | 1.41 | 1.00 | 1.73 | |
| 1.73 | 1.41 | 2.00 | 2.45 | 2.45 | 2.24 | 1.41 | 1.41 | 1.41 | 1.41 | 0.00 | 1.41 | 1.73 | 2.24 | |
| 1.73 | 1.41 | 1.41 | 1.41 | 2.00 | 1.73 | 1.41 | 1.41 | 2.00 | 1.41 | 1.41 | 0.00 | 1.73 | 1.00 | |
| 1.41 | 1.73 | 1.00 | 1.73 | 2.24 | 2.45 | 2.24 | 1.73 | 2.24 | 1.00 | 1.73 | 1.73 | 0.00 | 2.00 | |
| 2.45 | 2.24 | 1.73 | 1.00 | 1.73 | 1.41 | 1.73 | 2.24 | 2.65 | 1.73 | 2.24 | 1.00 | 2.00 | 0.00 | |

Below is my work.



B. Naive Bayes.

**1. Using the full dataset, induce the corresponding NB model. Show your result in the form of probability tables as we did in class.**

**2. What would your model predict for the following two Saturday mornings: <Oct 1, Overcast, Cool, High, Weak>, and <May 26, Sunny, Hot, Normal, Strong>? Show your work.**

Appendix

//===============================================================

// Name : DistanceMe.cpp

// Author : Christopher LaJon Morgan

// Version :

// Copyright : @clm

// Description : Hello World in C++, Ansi-style

//===============================================================

#include <iostream>

#include <math.h>

#include <iomanip>

using namespace std;

double distance(double a[4], double b[4])

{

return sqrt(pow(b[0] - a[0], 2) +

pow(b[1] - a[1], 2) +

pow(b[2] - a[2], 2) +

pow(b[3] - a[3], 2));

}

int main()

{

double data[][4] =

{

{2,2,1,0},

{2,2,1,1},

{1,2,1,0},

{0,1,1,0},

{0,0,0,0},

{0,0,0,1},

{1,0,0,1},

{2,1,1,0},

{2,0,0,0},

{1,1,0,0},

{2,1,0,1},

{1,1,1,1},

{1,2,0,0},

{0,1,1,1},

};

double answer[14][14];

memset(answer, 0, sizeof(double) \* 14 \* 14);

for (int i = 0; i < 14; i++)

{

for (int t = 0 + i; t < 14; t++)

{

answer[i][t] = distance(data[i], data[t]);

}

}

cout << fixed;

for (int r = 0; r < 14; r++)

{

for (int c = 0; c < 14; c++)

{

cout << setprecision(2) << answer[r][c] << " ";

}

cout << endl;

}

return 0;

}